30 DE MARZO DE 2009

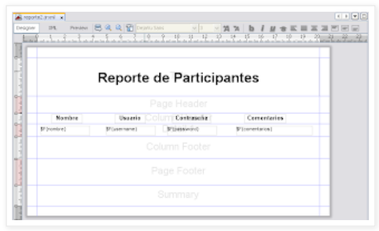
Creación de Reportes con JasperRepots y iReports - Parte 3: Parámetros y Variables

En el tutorial anterior vimos como hacer uso de los **`Field`**s para indicar la posición dentro de nuestro reporte en la que deben quedar los valores que obtenemos de nuestra aplicación Java.

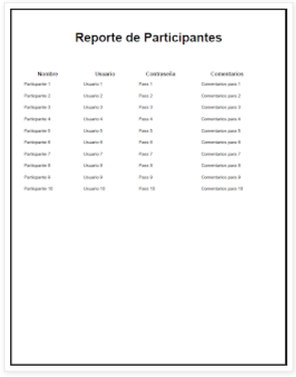
Ahora hablaré de los otros dos tipos de expresiones que proporciona **`JasperReports`**: **`Parameters`** y **`Variables`**. Y la forma en la que podemos usarlos para hacer más dinámicos nuestros reportes.

Para ello seguiremos usando el ejemplo del segundo tutorial , así que si no lo han leído les recomiendo que lo hagan en esta página.

Abrimos **`iReport`** y el reporte del tutorial anterior ("**`reporte2.jrxml`**"). En este momento nuestro reporte debe verse como en la siguiente imagen:



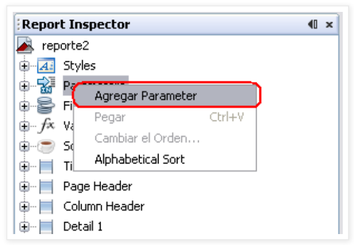
En donde solo tenemos textos estáticos y "**`fields`**" para mostrar un reporte básico de los datos de ciertos participantes. Si lo recuerdan, la salida del reporte al generarlo desde nuestra aplicación Java, era la siguiente:



Ahora pasaremos a agregar "**`Parameters`**" a este reporte. Como dije en el tutorial anterior: los parámetros son valores que usualmente se pasan al reporte desde el programa que crea el "**`JasperPrint`**" del reporte (nuestra aplicación Java) haciendo uso de un "**`java.util.Map`**".

En este caso pasaremos dos parámetros: el **título del reporte**, y el **nombre del autor** del mismo.

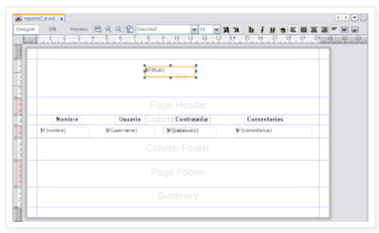
Al igual que con los **`Field`**s, para poder hacer uso de un parámetro primero hay que declararlo en el "**`Report Inspector`**" (situado en el panel izquierdo del **`iReport`**), por lo que nos dirigimos a esta ventana y hacemos clic derecho sobre el nodo "**`Parameters`**". Al hacer esto se abrirá un menú contextual. Seleccionamos la opción "**`Add Parameter`**" (la única habilitada).



Con esto se agregará un nuevo parámetro llamado "**`parameter1`**" de tipo "**`java.lang.String`**". Cambiamos en nombre del parámetro en la ventana de propiedades a "**`titulo`**" (recuerden que este nombre es el que deberemos usar desde nuestra aplicación Java para establecer el valor del parámetro).

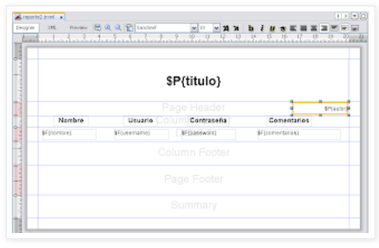
Agregamos otro parámetro llamado "**`autor`**" de la misma forma que agregamos el de "**`titulo`**".

Ahora eliminamos el elemento que teníamos como título del reporte (el que contiene el texto estático "**`Reporte de Participantes`**") y lo reemplazamos por el parámetro "**`titulo`**" arrastrándolo desde la ventana del "**`Report Inspector`**" a la banda "**`Title`**".



Ajustamos este elemento para que tenga el mismo formato que el título que teníamos anteriormente.

Ahora agregamos el segundo parámetro ("**`autor`**") en la banda "**`Page Header`**", la cual se repite en la parte superior de cada una de las páginas (en la primer página aparece después del título), arrastrándolo a esta banda.



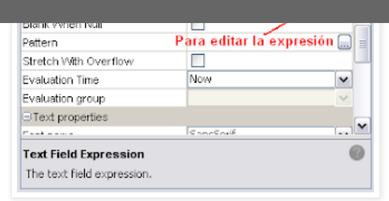
Si dejamos esta expresión tal y como está ("**`$P{autor}`**") solo se mostrará en nombre del autor en la parte superior de cada una de las páginas (por ejemplo si el autor del reporte es "**`Juan`**", solo veremos la palabra "**`Juan`**" en la parte superior de los reportes). Esto podría no ser claro para quienes revisen este reporte, por lo que queremos indicar de forma explícita que a lo que se refiere el valor de este parámetro es el nombre del autor. Para hacer esto nos dirigimos a la ventana de propiedades de este parámetro (para lo cual debemos tenerlo seleccionado) y buscamos la categoría "**`Text field properties`**". Ahi se encuentra una propiedad llamada "**`Text Field Expression`**" cuyo valor es "**`$P{autor}`**".

DATOS PERSONALES

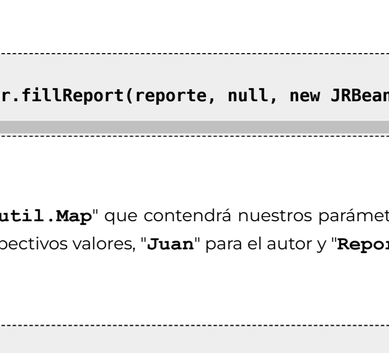
Alex

Programador Java con algunos años de experiencia en múltiples poyectos y con múltiples APIs y herramientas deseoso de compartir experiencias con el resto de programadores.

Ver todo mi perfil

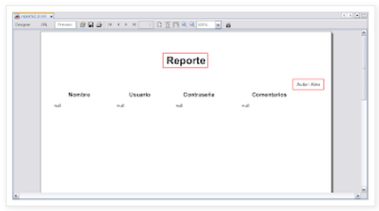


Lo que haremos es editar esta expresión para anteponer la palabra "**`Autor:  `**". Para hacer esto presionamos el botón "**`...`**", con lo que se abrirá una ventana que nos permitirá editar la expresión. En esta ventana modificamos la expresión "**`$P{autor}`**" para que quede **`"Autor:  " + $P{autor}`**. Con esto lo que logramos es que a la cadena estática "**`Autor:  `**" se le concatene el valor de la expresión dinámica "**`$P{autor}`**". Por lo que ahora el lo que aparecerá en la parte superior de cada página del reporte será la cadena "**`Autor: Juan`**".



Ahora presionamos el botón "**`Compile Report`**" para que se genere el reporte compilado "**`reporte2.jasper`**".

Veamos una vista previa del reporte, para esto hacemos clic en la pestaña "**`Preview`**". Nos pedirá los valores de los dos parámetros. Si colocamos para el parámetro "**`titulo`**" el valor "**`Reporte`**" y para el parámetro "**`autor`**" el valor "**`Alex`**" el preview que veremos será el siguiente:



Ahora veremos cómo pasarle estos valores desde nuestra aplicación Java, por lo que abrimos nuestro IDE de trabajo, **`NetBeans`**.

Ahí nos dirigimos a la clase "**`Main`**" que creamos la última vez y buscamos la línea en donde creamos nuestro objeto "**`JasperPrint`**":

```
JasperPrint jasperPrint = JasperFillManager.fillReport(reporte, null, new JRBeanCollectionDataSource(listaPariticipantes))
```

Y antes de esta línea crearemos un objeto "**`java.util.Map`**" que contendrá nuestros parámetros. Colocaremos el nombre de los parámetros (en este caso "**`autor`**" y "**`titulo`**") como llave y sus respectivos valores, "**`Juan`**" para el autor y "**`Reporte Participantes`**" para el título, como valores.

```
Map<String, String> parametros = new HashMap<String, String>();
parametros.put("autor", "Juan");
parametros.put("titulo", "Reporte Participantes");
```

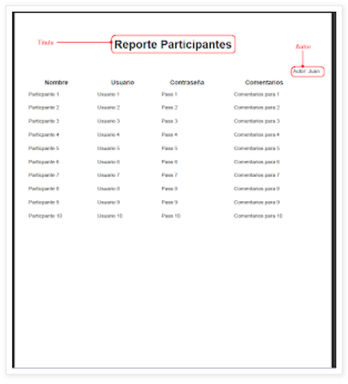
Y pasamos este "**`Map`**" como el segundo parámetro del método "**`fillReport`**", al que hasta ahora le hemos pasado "**`null`**":

```
JasperPrint jasperPrint = JasperFillManager.fillReport(reporte, parametros, new JRBeanCollectionDataSource(listaPariticipa
```

El código de nuestro método "**`main`**" queda así:

```
public static void main(String[] args) throws Exception
{
    List<Participante> listaPariticipantes = new ArrayList<Participante>();

    for (int i = 1; i <= 10; i++)
    {
        Participante p = new Participante(i, "Particpante " + i, "Usuario " + i, "Pass " + i, "Comentarios para " + i);
        listaPariticipantes.add(p);
    }

    JasperReport reporte = (JasperReport) JRLoader.loadObject("reporte2.jasper");

    Map<String, String> parametros = new HashMap<String, String>();
    parametros.put("autor", "Juan");
    parametros.put("titulo", "Reporte Participantes");

    JasperPrint jasperPrint = JasperFillManager.fillReport(reporte, parametros, new JRBeanCollectionDataSource(listaPariti

    JRExporter exporter = new JRPdfExporter();
    exporter.setParameter(JRExporterParameter.JASPER_PRINT, jasperPrint);
    exporter.setParameter(JRExporterParameter.OUTPUT_FILE, new java.io.File("reporte3PDF.pdf"));
    exporter.exportReport();
}
```

Si ejecutamos nuestro reporte obtendremos un archivo con el siguiente contenido:



Como podemos ver los parámetros "**`titulo`**" y "**`autor`**" han sido reemplazados por los valores "**`Reporte  Participantes`**" y "**`Juan`**" respectivamente.

Gracias a los parámetros podemos pasar valores (como en este caso cadenas, pero podemos pasar prácticamente cualquier tipo de objeto) a nuestro reporte directamente desde nuestra aplicación Java.

Ahora veremos el último tipo de expresión que **`JasperReports`** nos proporciona, las **Variables**.

Como dije en el post anterior: las variables son objetos usados para almacenar valores como los resultados de cálculos.

La segunda variable que usaremos será una variable creada por nosotros mismos y nos mostrará la suma de los puntos de los **`Participante`**s (en este momento los **`Participante`**s no tienen puntos, por lo que tendremos que agregarlos).

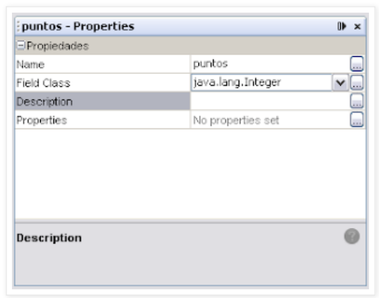
Lo primero que haremos es modificar la clase "**`Participante`**" agregando el atributo "**`puntos`**" de tipo "**`int`**" con sus correspondientes **`setters`** y **`getters`**:

```
private int puntos;

public int getPuntos()
{
    return puntos;
}

public void setPuntos(int puntos)
{
    this.puntos = puntos;
}
```

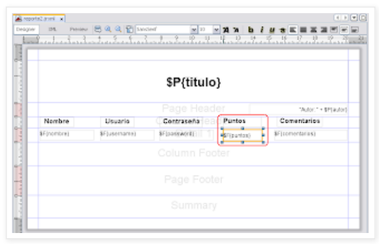
y en nuestro método "**`main`**" agregamos, en el ciclo que crea los objetos "**`Participante`**", el siguiente código para establecer los puntos de cada uno de los **`Participante`**s:

```
p.setPuntos(i);
```

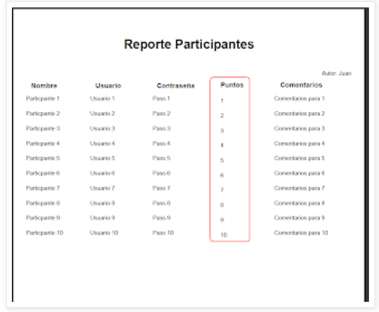
Ahora regresamos a **`iReport`** y agregamos un nuevo **`Field`** en la ventana "**`Report Inspector`**" llamado "**`puntos`**" de tipo "**`java.lang.Integer`**". Este **`field`** será usado para mostrar los puntos que tiene cada uno de los **`Participante`**s.



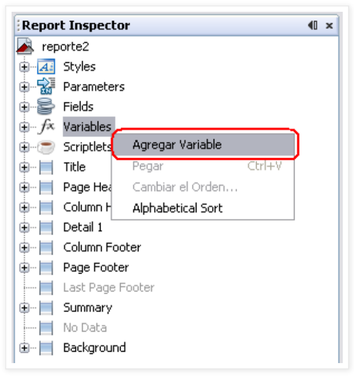
Agregamos una columna, dentro de la banda "**`Details 1`**", para mostrar este **`field`**:



Si compilamos nuestro reporte y ejecutamos nuestra aplicación Java debemos ver un reporte con el siguiente contenido:

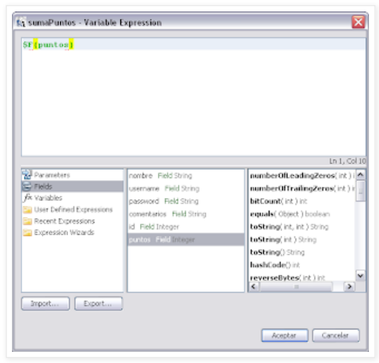


Ahora agregaremos la variable que realizará la suma de los puntos de todos los **`Participante`**s. Para esto agregamos una nueva variable en el nodo "**`Variables`**" de la ventana "**`Report Inspector`**" haciendo clic derecho sobre este nodo y seleccionando la opción "**`Add Variable`**" del menú contextual que se abre.

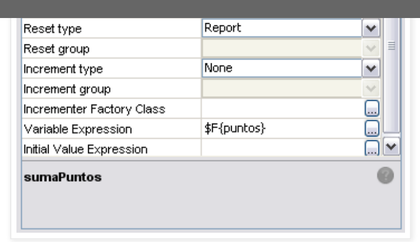


Modificamos sus propiedades en la ventana "**`Properties`**" cambiando su nombre a "**`sumaPuntos`**", su "**`Variable Class`**" a "**`java.lang.Integer`**", y "**`Calculation`**" a "**`Sum`**" (con esto indicamos que la variable mantendrá la suma de los valores que indicaremos más adelante, otras opciones incluyen un contador, el promedio, el valor más alto, etc.).

Ahora indicaremos la suma de cuáles valores son los que queremos que se almacene en este variable. Para eso hacemos clic en el botón "**`...`**" de la propiedad "**`Variable Expression`**", con lo que se abrirá una nueva ventana. Es esta nueva ventana indicaremos de dónde se tomarán los valores para la suma. Podemos usar una expresión tan compleja como necesitemos. Como en este caso solo queremos realizar la suma de los puntos escribimos "**`F{puntos}`**". Y presionamos el botón "**`OK`**" para confirmar.



Al final las propiedades de nuestra variable deben quedar así:



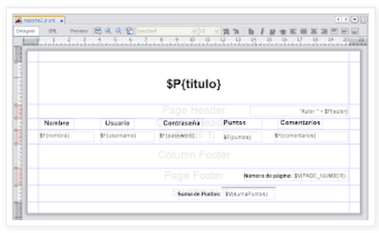
Ahora solo **arrastramos** nuestra variable "**sumaPuntos**" desde el "**Report Inspector**" hasta la banda "**Summary**". Esta banda se muestra cuando ya se han mostrado todos los datos de la banda "**Details 1**" y se muestra justo abajo de ella.

Agregaré un texto estático y una línea en esta banda solo para que se vea un poco mejor. Nuestro reporte hasta ahora debe verse así:

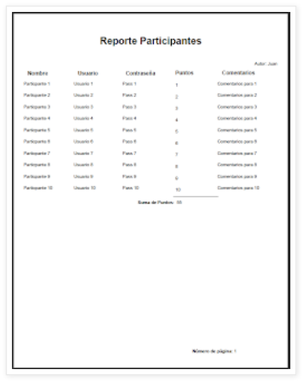
Ahora para terminar agregaremos la variable "**PAGE_NUMBER**" que nos dice el número de página actual en el reporte. Arrastramos esta variable desde el "**Report Inspector**" hasta la banda "**Page Footer**", la cual se muestra en la parte inferior de cada página.

Podría parecer un poco extraño que la banda "**Summary**" que se encuentra abajo de la banda "**Page Footer**" en el designer vaya a aparecer antes en el reporte final, pero confíen en mi, así será ^_^.

Agregaré también un texto estático solo para indicar que el valor que estamos mostrando corresponde al número de página. Al final el reporte queda así:



Ahora guardamos y compilamos nuestro reporte y ejecutamos nuestra aplicación desde el **NetBeans**. El reporte generado debe verse así:



Como podemos ver, la variable "**sumaPuntos**" contiene la suma de los puntos de todos los participantes (**55**).

Nuestra clase "**Participante**" queda de la siguiente forma:

```
public class Participante
{
    private int id;
    private String nombre;
    private String username;
    private String password;
    private String comentarios;
    private int puntos;


    public Participante()
    {
    }

    public Participante(int id, String nombre, String username, String password, String comentarios)
    {
        this.id = id;
        this.nombre = nombre;
        this.username = username;
        this.password = password;
        this.comentarios = comentarios;
    }

    public String getComentarios()
    {
        return comentarios;
    }

    public void setComentarios(String comentarios)
    {
        this.comentarios = comentarios;
    }

    public int getId()
    {
        return id;
    }

    public void setId(int id)
    {
        this.id = id;
    }

    public String getNombre()
    {
        return nombre;
    }

    public void setNombre(String nombre)
    {
        this.nombre = nombre;
    }

    public String getPassword()
    {
        return password;
    }

    public void setPassword(String password)
    {
        this.password = password;
    }

    public String getUsername()
    {
        return username;
    }

    public void setUsername(String username)
```

```
    {
        return puntos;
    }

    public void setPuntos(int puntos)
    {
        this.puntos = puntos;
    }
}
```

y nuestra clase "**Main**" de la siguiente forma (omitiendo los **import**s):

```
public class Main
{
    public static void main(String[] args) throws Exception
    {
        List<Participante> listaPariticipantes = new ArrayList<Participante>();
        for (int i = 1; i <= 10; i++)
        {
            Participante p = new Participante(i, "Particpante " + i, "Usuario " + i, "Pass " + i, "Comentarios para " + i)
            p.setPuntos(i);

            listaPariticipantes.add(p);
        }

        JasperReport reporte = (JasperReport) JRLoader.loadObject("reporte2.jasper");

        Map<String, String> parametros = new HashMap<String, String>();
        parametros.put("autor", "Juan");
        parametros.put("titulo", "Reporte Participantes");

        JasperPrint jasperPrint = JasperFillManager.fillReport(reporte, parametros, new JRBeanCollectionDataSource(listaPa

        JRExporter exporter = new JRPdfExporter();
        exporter.setParameter(JRExporterParameter.JASPER_PRINT, jasperPrint);
        exporter.setParameter(JRExporterParameter.OUTPUT_FILE, new java.io.File("reporte3PDF.pdf"));
        exporter.exportReport();
    }
}
```

Para terminar solo recuerden: los **parámetros** son expresiones que nos permiten pasar valores, desde nuestra aplicación Java, a los reportes generador por **JasperReports**. Y las **variables** almacenan valores que son el resultado de cálculos de distinta naturaleza sobre los valores que recibimos en nuestro reporte.

Bien, este es el final de este tercer tutorial sobre **JasperReports**. Espero que les sea de utilidad. No olviden dejar sus comentarios, dudas y sugerencias.

En el siguiente tutorial veremos cómo mostrar los reportes generados desde una aplicación web.

Saludos.

Descarga los archivos de este tutorial desde aquí:

- **Reporte con Parámetros y Variables**

Entradas Relacionadas:

- Parte 1: Reportes con Conexión a Bases de Datos
- Parte 2: Uso de DataSources Personalizados
- Parte 4: Reportes en aplicaciones web
- Parte 5: Gráficas en Reportes
- Parte 6: Grupos
- Parte 7: Subreportes

Labels: ireports, jasperreports, java, parametros, reportes, variables

Entrada más reciente Inicio Entrada antigua